

Acoustic modeling with mixtures of subspace constrained exponential models

Karthik Visweswariah, Scott Axelrod, Ramesh Gopinath

T. J. Watson Research Center, I.B.M
Yorktown Heights, NY, USA
{kv1, axelrod, rameshg}@us.ibm.com

Abstract

Gaussian distributions are usually parameterized with their natural parameters: the mean μ and the covariance Σ . They can also be re-parameterized as exponential models with canonical parameters $P = \Sigma^{-1}$ and $\psi = P\mu$. In this paper we consider modeling acoustics with mixtures of Gaussians parameterized with canonical parameters where the parameters are constrained to lie in a shared affine subspace. This class of models includes Gaussian models with various constraints on its parameters: diagonal covariances, MLLT models, and the recently proposed EMLLT and SPAM models. We describe how to perform maximum likelihood estimation of the subspace and parameters within a fixed subspace. In speech recognition experiments, we show that this model improves upon all of the above classes of models with roughly the same number of parameters and with little computational overhead. In particular we get 30-40% relative improvement over LDA+MLLT models when using roughly the same number of parameters.

1. Introduction

State of the art speech recognition systems use mixtures of Gaussians to model the acoustics x in each context dependent state s :

$$p(x|s) = \sum_{g \in \mathcal{G}(s)} \pi_g N(x; \mu_g, \Sigma_g), \quad (1)$$

where $x \in \mathbb{R}^d$ and $\mathcal{G}(s)$ is the set of Gaussians modeling state s . Usually the covariances of these Gaussians are assumed to be diagonal for robust parameter estimation, compact representation and to allow fast evaluation. Various methods of tying parameters of a set of Gaussians have been shown to improve modeling performance: MLLT (a.k.a semi-tied covariances) [1], [2], EMLLT [3], and SPAM ([4]). In MLLT the precision (inverse covariance) of a Gaussian g is modeled as $A\Lambda_g A^T$ where A (shared across all Gaussians) and Λ_g (which is diagonal) both are $d \times d$ square matrices. EMLLT also models the precisions as $A\Lambda_g A^T$, but generalizes to allow A to be non-square. A has size $d \times D$ and Λ_g is diagonal and has size $D \times D$. In EMLLT the precisions are restricted to lie in a subspace spanned by $D > d$ rank one matrices. In SPAM one considers Gaussians as exponential distributions with canonical parameters $P = \Sigma^{-1}$ and $\psi = P\mu$. SPAM [5] restricts the precisions P_g to lie in an affine subspace, and further [4] restricts ψ_g to lie in an affine subspace.

In this paper we consider a further generalization of SPAM. Like in SPAM we restrict the canonical parameters ψ and P to lie in an affine subspace. However unlike SPAM we do not treat these two parameters differently and they are together restricted to lie in a subspace. In this paper we describe how to train this subspace and the coefficients to maximize likelihood. We show

that a model with this general structure improves significantly over SPAM models especially when the dimension of the subspace is small. The techniques used to train the subspaces here also allow us to train SPAM subspaces to maximize likelihood. This improves the performance of SPAM models as compared to the results reported in [4] where the subspaces were trained using a quadratic approximation to the likelihood.

The outline of the rest of the paper is as follows. In Section 2 we parameterize Gaussians as exponential models and make precise the structure of the model we consider and its relation to other previously studied models. Section 3 describes maximum likelihood parameter estimation for these models. In Section 4 we present our experimental results. Section 5 presents our conclusions.

2. Model structure

We first re-parameterize Gaussian distributions using its canonical parameters as an exponential distribution. A multivariate Gaussian (parameterized by μ and Σ) is given by

$$N(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp -1/2(x - \mu)^T \Sigma^{-1} (x - \mu).$$

Letting $P = \Sigma^{-1}$ and $\psi = \Sigma^{-1}\mu$ we can rewrite this as

$$N_e(x; \psi, P) = \frac{1}{Z(\psi, P)} \exp \left(-1/2x^T P x + \psi^T x \right),$$

where

$$\log Z(P, \psi) = d/2 \log 2\pi - 1/2 \log |P| + 1/2 \psi^T P^{-1} \psi$$

ensures that the density integrates to one. Note that $-1/2x^T P x + \psi^T x$ is linear in the parameters (ψ, P) and thus it is clear that the Gaussian is in the class of exponential distributions, as described as follows.

We first give some background about distributions in the exponential family. Exponential distributions on $x \in \mathbb{R}^d$ parameterized by $\theta \in \Theta(f)$ are distributions of the form $\exp(\theta^T f(x))/Z(\theta, f)$. $Z(\theta, f) = \int_x \exp \theta^T f(x) dx$ is a normalizing constant. $\Theta(f)$ is the set of θ for which $Z(\theta, f)$ is defined and is the allowed range of parameters. Exponential distributions have several well known properties, e.g. for a given f , $\log Z(\theta, f)$ is convex and the set $\Theta(f)$ is convex. In evaluating a large number of distributions in an exponential family with a given f the computational cost is divided into computing $f(x)$ for a given feature vector and computing the inner product of $f(x)$ with the parameters for each distribution. The latter cost dominates when the set of distributions is large. We note that if we restrict $\theta = B\lambda$ to lie in a subspace (given by B) we still get distributions in the exponential family. If the dimension of the subspace is much smaller than the dimension of the

original space we can get significant reductions in computation at the additional cost of computing $B^T f(x)$. If the number of distributions evaluated is large as is typical in speech recognition systems the computational savings can be significant. Of course this decrease in computation results in weaker models. Using this framework of restricting the canonical parameters of an exponential to a subspace allows for a smooth tradeoff between computational cost and modeling power. In particular for Gaussians we would not get any savings in evaluation time if we restricted the natural parameters (means and covariances) to a subspace. Although we focus on Gaussians in this paper the theory and the algorithms used work even for other exponential models, provided: (a) we can calculate $Z(\theta, f)$ and its gradient with respect to θ efficiently; (b) for $\theta_0 \in \Theta$ and any $\theta_1 \in \Theta$ we can calculate the maximum step size t such that $\theta_0 + t\theta_1 \in \Theta$. We will see in Section 3 that our estimation algorithms only depend on this.

In the case of the Gaussian $f(x)^T = [x^T - 1/2 \text{vec}(xx^T)^T]$ and $\theta^T = [\psi^T \text{vec}(P)^T]$, where ψ is d dimensional and $\text{vec}(P)$ is the free elements of the matrix P written as a vector in some fixed order with the off diagonal elements multiplied by $\sqrt{2}$. Since P is symmetric it has $d(d+1)/2$ free elements. The reason that we multiply the off diagonal terms by $\sqrt{2}$ is to maintain inner products. Thus for two symmetric matrices S_1 and S_2 $\text{trace}(S_1 S_2) = \text{vec}(S_1)^T \text{vec}(S_2)$. The constraint that Gaussians have positive definite covariances translates into P being positive definite. We will interchangeably use (ψ, P) or θ as is convenient.

Various methods of parameter tying across Gaussians like MLLT, EMLLT and SPAM are methods of restricting one or both of ψ, P to be in a subspace (possibly affine) as given by:

$$\begin{aligned} \theta_g &= \begin{bmatrix} \psi_g \\ \text{vec}(P_g) \end{bmatrix} \\ &= \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \lambda_g + \begin{bmatrix} \psi_0 \\ \text{vec}(P_0) \end{bmatrix}. \end{aligned}$$

B_{11} is a $d \times L$ matrix, B_{12} is a $d \times D$ matrix, B_{21} is a $d(d+1)/2 \times L$ matrix and B_{22} is a $d(d+1)/2 \times D$ matrix. In MLLT, EMLLT and in SPAM $B_{12} = 0$, $B_{21} = 0$. Furthermore, in EMLLT and in MLLT B_{11} is the identity matrix, $\psi_0 = 0$, $P_0 = 0$, and the columns of B_{22} correspond to rank-one matrices ($\text{vec}(aa^T)$). In MLLT B_{22} has exactly d columns where as in EMLLT it has more than d columns.

In this paper we consider the case when $\theta_g = B\lambda_g$ and there is no arbitrary structure imposed on B . Although it would be easy to consider the case where B is shared across classes of Gaussians in this paper we assume that the parameters in B are tied across all Gaussians. We note that the structure of B affects the precomputation cost; we want structures that balance the precomputation cost and performance for a given basis size. We show that placing no structure provides an improvement in performance for the same basis size. Although we do not investigate it here, we could place some structure as in the hybrid models described in [6] which reduces precomputation cost without affecting performance.

The model we use is a mixture of exponential distributions (Gaussians) for each context dependent state s :

$$P(x|s) = \sum_{g \in \mathcal{G}(s)} \pi_g N_e(x; B_1 \lambda_g, B_2 \lambda_g),$$

where

$$\begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = B,$$

and B_1 has d rows and B_2 has $d(d+1)/2$ rows. Through out this paper we assume that the Gaussians are not shared across states. This assumption is made for simplicity of exposition; all the methods to estimate parameters work even with sharing of Gaussians across states.

3. Maximum likelihood parameter estimation

In this section we describe algorithms that we use to estimate mixtures of Gaussians with the canonical parameters restricted to be in a subspace. We describe how to estimate the subspace B and the parameters $\{\lambda_g\}$.

Let $\Omega = (\{\lambda_g, \pi_g\}, B)$. Assume we have training data (x_t, s_t) (each feature vector is labeled with the state) then the log-likelihood of the training data is

$$L(\Omega) = \sum_t \log \sum_{g \in \mathcal{G}(s_t)} \pi_g N_e(x_t; B_1 \lambda_g, B_2 \lambda_g).$$

Although we carry out all our optimization using well known generic numerical optimization techniques we cannot directly optimize the log-likelihood as each function evaluation would involve running through all the training data. As always, we resort to the EM algorithm. The E-step is standard, the M-step has no closed form solution and we use numerical techniques to solve the optimization problem. The Q function that we need to optimize over Ω is:

$$\begin{aligned} Q(\Omega, \hat{\Omega}) &\simeq \sum_s N(s) \sum_{g \in \mathcal{G}(s)} \left(-\tilde{\pi}_g \log Z(P_g, \psi_g) \right. \\ &\quad \left. + \tilde{\pi}_g \theta_g^T \langle f(x) \rangle_g + \tilde{\pi}_g \log \pi_g \right). \end{aligned} \quad (2)$$

In the above equation

$$\log Z(P, \psi) = -1/2 \log(\det P) + 1/2 \psi^T P^{-1} \psi,$$

$$\tilde{\pi}_g = \frac{1}{N(s(g))} \sum_{t: s_t = s(g)} \gamma_t(g),$$

$$\gamma_t(g) = \gamma_t(g, \hat{\Omega}) = \frac{\hat{\pi}_g P(x_t | g, s(g), \hat{\Omega})}{p(x_t | s(g), \hat{\Omega})}$$

for t such that $s_t = s(g)$ and zero otherwise,

$$\langle f(x) \rangle_g = \frac{1}{n(g)} \sum_{t: s_t = s(g)} \gamma_t(g) f(x_t).$$

Here $s(g)$ is the HMM state that Gaussian g belongs to, $N(s)$ is the number of samples associated with state s and $n(g) = N(s(g))\tilde{\pi}_g$. Although we haven't written it out explicitly, in the RHS of (2) $P_g = B_2 \lambda_g$, $\psi_g = B_1 \lambda_g$ and $\theta_g^T = [\psi_g^T \text{vec}(P_g)^T]$ are functions of λ_g and B . Note also that though we described the E-step in a Viterbi framework, everything works similarly if the posteriors are calculated using the forward-backward algorithm.

Optimizing the Q function over π_g is easy and results in $\pi_g = \tilde{\pi}_g$. Substituting in the optimal parameter values for π_g the function that we are left with is:

$$Q(\{\lambda_g\}, B) = \sum_g -n(g) \left(\log Z(P_g, \psi_g) - \theta_g^T \langle f(x) \rangle_g \right) \quad (3)$$

To optimize over B and $\{\lambda_g\}$ we adopt a round robin strategy. We fix one and optimize over the other and alternate this procedure. We note that $\log Z(\psi, P)$ is convex and the domain of Z is a convex set (using the properties of exponential models). Also $\{\psi_g\}$, $\{P_g\}$ and $\{\theta_g\}$ are linear in each of B and $\{\lambda_g\}$ when the other is fixed. This implies that when we hold fixed one of B , $\{\lambda_g\}$, the optimization problem with respect to the other is convex with a unique global maximum. Although each step has a unique maximum there is no guarantee that we have a global maximum for our problem. For this reason and to limit the run time it is essential to choose the starting parameter values intelligently.

3.1. Parameter initialization

We initialize our parameter values by a SPAM basis (i.e we hold B_{12} and B_{21} fixed at zero) using the methods described in [4]. Assuming we want a subspace of dimension n we first find a precision basis B_{22} of size $n/2$ using the method in [5] which makes a quadratic approximation to the likelihood. We next find precision coefficients in this basis following [5]. B_{11} is also initialized by first solving an approximation to the Q function that results in a closed form solution. With this initialization we alternate between updating B_{11} and the coefficients in this basis. Each of these problems is a quadratic optimization problem with a closed form solution. We note that this initialization procedure is much faster than the training that follows.

3.2. Parameter updates

We use the limited memory BFGS algorithm [7] along with the More-Thuente line search algorithm [8] as implemented in the open source package [9] to perform parameter estimation. This package is used with a slight modification to allow fast line searches. The limited memory BFGS algorithm finds search directions by trying to approximate the Hessian. All that is required is to evaluate the function and its gradient reasonably quickly.

We first consider optimizing over $\{\lambda_g\}$. From (3) it is clear that to optimize over λ_g for a particular Gaussian g we can ignore all terms in the sum except one. Thus we can perform the optimization over $\{\lambda_g\}$ in parallel and independently for each Gaussian. We drop the index g and write

$$Q(\lambda) = -\log Z(P, \psi) + \theta^T \langle f(x) \rangle .$$

Clearly the cost of computing the function is dominated by the cost of computing $Z(P, \psi)$ which requires calculating the inverse and the determinant of P . We now focus on the gradient. Let $\nabla_x Q$ denote the gradient of Q with respect to some variables x . Then

$$\nabla_\psi Q = -P^{-1}\psi + \langle x \rangle$$

and

$$\nabla_P Q = 1/2 \left(P^{-1} + (P^{-1}\psi)(P^{-1}\psi)^T - \langle xx^T \rangle \right) .$$

Using the chain rule we have the gradient of Q with respect to λ :

$$\nabla_\lambda Q(\lambda) = B_1^T \nabla_\psi Q + B_2^T \text{vec}(\nabla_P Q) .$$

We now turn to computations required to optimize over B . Since B is tied across all Gaussians, in optimizing B , each function and gradient evaluation requires a sum over all Gaussians. Since we implement this sum without parallelization, the optimization over B is the bottleneck in our implementation. The

function is evaluated using (3). The gradient with respect to B is:

$$\nabla_B Q(B) = \sum_g n(g) \left[\begin{array}{c} \nabla_{\psi_g} Q \\ \text{vec}(\nabla_{P_g} Q) \end{array} \right] \lambda_g^T .$$

The optimization technique we use works by searching along a line from the current point at each iteration, i.e. we need to find maxima of $Q(\lambda + t\Delta\lambda)$ and $Q(B + t\Delta B)$ with respect to t when optimizing over λ and B respectively. Since the domain of our Q function is constrained to P_g being positive definite for each Gaussian; we need to restrict the maximum step size to ensure that we stay within the domain during each line search. Since P is linear in each of λ and B fixing the other, this amounts to finding the largest value t such that $P + t\Delta P$ is positive definite (given P is positive definite). This largest t can be found as in [4] as follows. By an eigen analysis of $P^{-1/2}\Delta P P^{-1/2}$ we find diagonal D and orthonormal E such that $P^{-1/2}\Delta P P^{-1/2} = EDE^T$. Then $P + t\Delta P = P^{1/2}E(I + tD)E^T P^{1/2}$. Thus $P + t\Delta P$ and $I + tD$ are positive definite for the same values of t , which makes it easy to find the allowed values of t . For optimization over B the maximum step size is found by taking the minimum over the (maximum) step sizes for all Gaussians. The computation performed in limiting the line search can also be used to speed up evaluation of the function along a line, which helps speed up the line searches that are performed at each iteration of the optimization. Since both P and ψ are linear in each of B and $\{\lambda_g\}$ given the other fast evaluation of $Q(\lambda + t\Delta\lambda)$ or $Q(B + t\Delta B)$ for different t reduces to evaluating $Q(P + t\Delta P, \psi + t\psi)$. Looking at the Q function given by (3) it is clear that the second term can be evaluated quickly along a line by storing $\theta^T \langle f(x) \rangle$ and $\Delta\theta^T \langle f(x) \rangle$. The first term in (3) can be evaluated quickly along a line as follows:

$$\begin{aligned} & -2 \log Z(P + t\Delta P, \psi + t\psi) = \\ & \log |P + \Delta P| - (\psi + \Delta\psi)^T (P + \Delta P)^{-1} (\psi + \Delta\psi) \simeq \\ & \sum_{i=1}^d \log(1 + tD_{ii}) - \frac{((v + t\Delta v)_i)^2}{1 + tD_i} , \end{aligned}$$

where $v = E^T P^{-1/2}\psi$, $\Delta v = E^T P^{-1/2}\Delta\psi$.

Once the basis is trained as described above we train the coefficients within the basis via EM. In the procedure for training B described above the statistics we needed were $\{\langle f(x) \rangle_g\}$, i.e. full covariance statistics. When we just need to update the untied parameters the set of statistics we need is $\{\langle B^T f(x) \rangle_g\}$, which is much smaller and hence easier to collect.

Although we described the training of an unconstrained basis; it is easy to put linear constraints on the basis with minor modifications. In particular it is easy to restrict B_{12} and B_{21} to be zero, this is how we trained SPAM bases for models presented in Section 4.

4. Experimental setup and results

All experiments reported on in this paper were conducted on a test database collected in a car [3]. We report word error rates on a test set comprised of small vocabulary grammar based tasks (addresses, digits, command and control) and consists of 73743 words. Data for each task was collected at 3 speeds: idling, 30mph and 60mph. All acoustic models (except for LDA+MLLT models) reported on here were built on $d = 52$

dimensional feature vectors. The 52 dimensional vector was obtained by projecting down from nine cepstral vectors (each thirteen dimensional) spliced together. The projection was obtained using LDA. The size $d = 52$ was chosen because the best full-covariance model performance was obtained at this size [4]. The acoustic model used separate digit phones with a total of 89 phones. All the acoustic models had a total of 10253 Gaussians distributed across 680 context dependent states using BIC based on a diagonal covariance system.

For all experiments, acoustic models were built from a fixed Viterbi alignment of the training data using a baseline diagonal covariance model. To train the basis, we first collected the statistics $\{n(g), \langle f(x) \rangle_g\}$ using a full covariance model. Once the basis is trained we then train models in that fixed basis via the EM algorithm from the aligned data.

The cost of training the basis depends on the size of the basis, the number of Gaussians and the dimension of the space. For $d = 52$, with the 10253 Gaussians that we have here, the training took about one to two days for the basis sizes we considered. Note that this time depends on the convergence stopping criterion and also on the speed of the underlying library used to perform the key matrix computations (especially the inversions and the generalized eigen value computations). We used the freely available library ATLAS [10] and also IBM's ESSL [11]. When the number of Gaussians in a system is much larger we could reduce the computational load by using one Gaussian per state for training the basis.

Table 1 presents our experimental results. The first column is the number of parameters per Gaussian. All systems in the same row have the same number of untied parameters (and thus roughly the same total number of parameters since the tied parameters contribute a small fraction). Results in column SPAM0 and SPAM1 in Table 1 correspond to the SPAM models of the same structure; in each row with n parameters $D = L = n/2$. They differ in that SPAM0 bases were trained as described in [4] whereas SPAM1 bases were trained to maximize likelihood. Models in column ExpModel are exponential models with the canonical parameters in a subspace of size n . All parameters of these models are trained to maximize likelihood starting from models in column SPAM0. We note that, for small basis sizes, significant gains result from allowing the generality of subspace constrained exponential models. We also note that SPAM models significantly outperform LDA+MLLT models; more so than reported in [4] due to the improved basis training algorithm used here. One more baseline number is the 52 dimensional LDA+MLLT model that has a WER of 2.7%. From Table 1, we see that this is obtained using a ExpModel with 26 parameters per Gaussian i.e with about 25% of the parameters of the LDA+MLLT model.

| Size | LDA+MLLT | SPAM0 | SPAM1 | ExpModel |
|------|----------|-------|-------|----------|
| 16 | - | - | 4.69 | 3.42 |
| 26 | 4.92 | 4.12 | 3.26 | 2.71 |
| 40 | 3.71 | 3.06 | 2.59 | 2.35 |
| 78 | 2.85 | 2.13 | 2.10 | 1.95 |

Table 1: WER comparison of LDA+MLLT, SPAM0, SPAM1, ExpModel

In Table 2 we see results comparing various SPAM models with 26 parameters per Gaussian to the more general exponential model. We see first of all (perhaps surprisingly) that is better to use more parameters to model the precision subspace. We also note that the exponential model outperforms all the SPAM

models significantly. At any rate it obviates the need to optimize over D for fixed $D + L$.

| Model Type | WER |
|-----------------|------|
| SPAM(L=18,D=8) | 3.52 |
| SPAM(L=13,D=13) | 3.26 |
| SPAM(L=8,D=18) | 3.00 |
| Exp(26) | 2.71 |

Table 2: WER comparison of LDA+MLLT, SPAM0, SPAM1, ExpModel

5. Conclusions

Parameterizing Gaussians as exponential models and using models where the parameters of these exponential models are constrained to be in a subspace provides a flexible framework where one can optimize WER at any given computational cost. These models provide large improvements over LDA+MLLT models and significant improvement over SPAM models of the same size (30% to 40% relative depending on the model size). Another way of characterizing these gains is that in one instance we get the same performance with roughly a quarter of the parameters. As a final note, we have seen that training the SPAM bases by maximizing likelihood gives significant gains (0% to 20% depending on the basis size).

6. References

- [1] R. Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," *Proceedings of ICASSP*, 1998.
- [2] M. J. F. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions in Speech and Audio Processing*, 1999.
- [3] P. Olsen, R. Gopinath, "Modeling inverse covariance matrices by basis expansion," *Proceedings of ICASSP*, 2002.
- [4] S. Axelrod, R. Gopinath, P. Olsen, K. Visweswariah, "Dimensional reduction, covariance modeling and computational complexity in asr systems," *Proceedings of ICASSP*, 2003.
- [5] S. Axelrod, R. Gopinath, P. Olsen, "Modeling with a subspace constraint on inverse covariance matrices," *Proceedings of ICSLP*, 2002.
- [6] K. Visweswariah, P. Olsen, R. Gopinath, S. Axelrod, "Maximum likelihood training of subspaces for inverse covariance modeling," *Proceedings of ICASSP*, 2003.
- [7] D. C. Liu, J. Nocedal, "On the limited memory bfgs method for large scale optimization problems," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.
- [8] More, Thuente, "Line search algorithms with guaranteed sufficient decrease," *ACM TOMS*, vol. 20, no. 3, pp. 286–307, 1994.
- [9] M. S. Gockenbach, W. W. Symes, "The Hilbert Class Library," <http://www.trip.caam.rice.edu/txt/hclldoc/html/>.
- [10] R. Clint Whaley, Antoine Petitet, et. al., "Automatically Tuned Linear Algebra Software," <http://math-atlas.sourceforge.net/>.
- [11] "Engineering Scientific Subroutine Library," www.rs6000.ibm.com/software/Apps/essl.html.